such as the following: attachment of lifecycles, starting workflows, sending notifications, auto-attribution, access control list assignment, custom security assignment, retention policy assignment, other custom-defined actions. In some embodiments, at instantiation, the smart container template provides a default configuration. If permissions and policies allow, these defaults can be overridden for the new instance. In some embodiments, the instance is saved as a new template. In some embodiments, if defaults have been overridden during instantiation or if changes have been made for the instance that was started using a template, users will have the option to save these changes to the source template or to save them to a new template. In **906**, instantiation is completed. Instantiation is complete after all instantiation actions and policies have successfully run, and the instance model has been flagged as a "runtime" model. In some embodiments, all instantiation related actions are logged.

[**0025**] FIG. **10** is a flow diagram illustrating an embodiment of a process for applying a smart container policy during runtime. In **1000**, an object is contributed to a smart container. Contribution to a Smart Container may happen through user interactions or through automated processes. Objects may be contributed one by one or through high ingestion operations. In various embodiments, users may explicitly link objects into a folder that is part of the smart container instance; an object may be auto-categorized, based on its content and/or attribute values to a folder that is part of the smart container instance; an object may be associated with a root-level smart container object that has a policy which further-categorizes the object and places the document in the appropriate spot within the smart container structure; or an object may be created as a member of a smart container during instantiation—the nodes of the model become members of the smart container at runtime.

[**0026**] In **1002**, a policy is executed. A policy is a discrete bit of business logic that is defined to execute based on some event or object operation. It may be implemented as an aspect attached to an object, such as by modifying an instance of an implementation object to add to that instance one or more methods or attributes not included in every object of that type; as a type-based behavior implemented by modifying or extending a base implementation object to include additional type-based behaviors; or script. In some embodiments, out-of-the-box smart containers will include several common policies and will provide a framework for creating other policies. All users and programmatic manipulation of an object in the managed content system will honor the policies that have been applied to that object. Smart container configuration associates policies explicitly—for example, a policy is associated directly with the object configuration or object class; a policy is associated with a relation type and the object is referenced by a relationship of that type; or a policy is inherited by the object from a parent object. A policy could have been associated with an object in a number of ways in addition to those associated explicitly through smart container configuration—for example, a policy is associated with the user object, role, or group; a policy associated with an object type; a policy is associated with the document template that a document was created from; a policy related to format; or a policy that is programmatically applied to an object or is user selected. A policy has a number of elements including a trigger which defines when a policy executes, a declarative or code-based set of conditions that will be evaluated at execution time to true or false, and an outcome (e.g., positive, negative, and error). A policy can manage conflicts. In the case where multiple policies may be associated with an object either directly or through inheritance, the system detects that there is a policy conflict (e.g., an object has multiple policies of the same type—for example two security or two versioning policies). The appropriate conflict resolution technique is applied including include 1) one policy takes precedence and only that policy is applied, 2) policies are applied sequentially, or 3) an error is indicated because the conflict cannot be resolved.

[**0027**] In **1004**, interactions are executed. At runtime, users and programs interact with a smart container and its contents. Interactions include viewing and operating on objects controlled by the smart container configuration. Viewing will display information regarding a smart container object in a manner as indicated by the viewing policy. For example, email is viewed in a split window display with one window containing in box, out box, and junk box folders, another window listing the recent inbox emails, and another window showing the contents of a particular email. Views can be different dependent on the object, the user, the role of the user, the security policy, or any other appropriate criteria. In various embodiments, operations for smart containers include one or more of the following: fill a placeholder with an actual document object; attach a policy to an object; remove a policy from an object; add a document to an smart container; remove a document from an smart container; freeze smart container; version smart container; lock/unlock smart container; export smart container; copy smart container; check-out/check-in smart container; cancel smart container checkout; replicate smart container; and/or make smart container reference. Also, objects governed by a smart container may be operated on in any of the usual ways, i.e. copying, moving, versioning, saving, get file, set file, set attribute, get attribute, lifecycle operations, attach to workflow, creation, deletion, etc. However, because the object is governed by a smart container configuration, these operations may be restricted, modified, overridden, or extended in some way by the policies on the object.

[**0028**] Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method for managing content comprising:

receiving a definition of a logical structure configured to manage associated content; and

applying to an item of content associated with an instance of the logical structure a policy or operation specified by the definition.

2. A method as in claim 1, further comprising creating the instance of the logical structure.

3. A method as in claim 1, further comprising creating the instance of the logical structure using a template.

4. A method as in claim 1, further comprising creating the instance of the logical structure using a template wherein the template is configure to propagate changes to the template by one of the following: to propagate template changes to